W-IE–NE-R

OPC Server for W-IE-NE-R

Crate Remote Control

User's Manual

00653.A0

General Remarks

The only purpose of this manual is a description of the product. It must not be interpreted as a declaration of conformity for this product including the product and software.

W-Ie-Ne-R revises this product and manual without notice. Differences of the description in manual and product are possible.

W-Ie-Ne-R excludes completely any liability for loss of profits, loss of business, loss of use or data, interrupt of business, or for indirect, special incidental, or consequential damages of any kind, even if **W-Ie-Ne-R** has been advises of the possibility of such damages arising from any defect or error in this manual or product.

Any use of the product which may influence health of human beings requires the express written permission of **W-Ie-Ne-R**.

Products mentioned in this manual are mentioned for identification purposes only. Product names appearing in this manual may or may not be registered trademarks or copyrights of their respective companies.

Table of contents:

1 General Description

OPC (**O**LE for **P**rocess **C**ontrol) allows fast and secure access to data and information under Windows operating systems. As an industry-spanning, multi-vendor software interface, OPC minimizes connection and maintenance overheads.

This server, running on a Computer with the Microsoft Windows 2000 operating system, enables access to all power supplies which are connected to the computer's CAN network card(s). It is possible to

- access from any OPC Client application to the data of one or more servers
- encapsulating the properties specific to the server and type of communication
- restricting access rights by the underlieing Microsoft DCOM.

2 Installation and Removal of the Software

2.1 Installation

2.1.1 WienerOPC Installation

- Unpack the file "OPC1005.zip" to a local directory (e.g. c:\opcserver). Then the directory contains the files:

-

WienerOPC_KVASER.exe	The server executable
SOSrvas.dll, SOCmnas.dll, SODaSas.dll	The OPC Toolbox libraries
WienerConfig.exe	Utility to setup the server configuration file (optional).
Wiener_cfg.xml	The sample server configuration file
OPC Server for WIENER Devices.doc	

- From a command prompt, execute the command:

  c:\opcserver\WienerOPC /RegServer

  or

  c:\opcserver\WienerOPC -RegServer

  This generates the necessary registry entries. (The full path is mandatory, the command line switches are not case-sensitive).

  There is a possibility to enable tracing internal error, warning, info and debug messages into two log-files: Wiener_log1.txt and Wiener_log2.txt. (refer to Tracing Options section).

- Some OPC Clients requires to run registration command on their computer even if OPC Server will run on the remote PC. The application specific Registry Key Entries are created during this operation, and OPC Server executables can be removed then, if OPC Server has to be run remotely.

2.1.2 OPC Server Browser Installation

You need to have installed "OPCEnum.exe" application on computer where OPC servers are to be accessed (any OPC DA2.0 client uses it to browse a list of available OPC servers on that machine). It needs the proxy/stub which is contained in "opccomn_ps.dll" library.

The OPC Common Proxy/Stub opcproxy.dll have to be installed for remote OPC Client-Server connection.

- Check if OPCEnum.exe, opccomn_ps.dll and opcproxy.dll are copied to the main WINDOWS directory (%SystemRoot%\system32)
- Copy them from "OPC Common\OPC" folder, **if not found**. **Be sure not to overwrite any newer versions** . Both of them need to be registered in your system by running (from menu Start, Run..) the command line:

  "%SystemRoot%\system32\ opcenum.exe" /RegServer

  or

  "%SystemRoot%\system32\ opcenum.exe" /Service

  (to install server browser as a service)

and

REGSVR32 OPCComn_ps.dll

REGSVR32 opcproxy.dll

- *Check to see if actprxy.dll is present and installed on your system. Most, but not all, NT systems will have this installed. If it is not present review the licence information and run APRXDIST.EXE from* "OPC Common\actprxy" *to install the needed software. This would potentially need to be installed on all server and client machines*

Refer to "OPC Server for W-IE-NE-R Crate Remote Control Definitions and Interfaces" document from OPC Foundation for OPC common software installation.

2.2 Trace Options

Trace messages of different levels and from different OPC Server threads can be logged into two logfiles: Wiener_log1.txt and Wiener_log2.txt (each of them 1MB capacity at last).

To enable tracing, and set trace levels options:

- From a command prompt, execute the command:

  c:\opcserver\WienerOPC */option1 /option2 … /optionn*

  or

  c:\opcserver\WienerOPC *-option1 -option2 …-/optionn*

  option switches are not case sensitive.

- The following options are available:

TraceOn	- enables tracing
TraceOff	- disables tracing
AllErrors	- enables all error messages (including internal OPC Toolkit's)
OPCErrors	- enables error messages from OPC level
WienerErrors	- enables Wiener specific errors
NoErrors	- disables error tracing
AllWarnings	- enables all warning messages (including internal OPC Toolkit's)
OPCWarnings	- enables warning messages from OPC level
WienerWarnings	- enables Wiener specific warnings
NoWarnings	- disables warning tracing
AllInfo	- enables all info messages (including internal OPC Toolkit's ones)
OPCInfo	- enables info messages from OPC level
WienerInfo	- enables Wiener specific infos
NoInfo	- disables info tracing
AllDebug	- enables all debug messages (including internal OPC Toolkit's)
OPCDebug	- enables debug messages from OPC level
WienerDebug	- enables Wiener specific debug messages

NoDebug - disables debug tracing

All tracing settings are kept in the Registry Keys and are read during application initialization.

Debug and Info options should be disabled during normal runtime, as they are time and memory consuming (especialy OPC Toolkit's tracing).

2.3 Cache Namespace Items

There are 7 Cache R/W items representing all datatypes used in Wiener OPC Server Namespace (created for OPC compliance tests purposes). User can use that Items to keep an information defined by himself in there.

2.4 Deinstallation

- From a command prompt, execute the command:

  c:\opcserver\WienerOPC /UnregServer

  or

  c:\opcserver\WienerOPC -UnregServer

  This deletes the used registry entries. (The full path is mandatory, the command line switches are not case-sensitive)
- Delete all files of the local directory (e.g. c:\opcserver).

2.5 Used Registry Key Entries

HKEY_CLASSES_ROOT\CLSID\{440325D6-6779-48D0-970F-02C9C7AAAAD2}

-with subkeys

HKEY_CLASSES_ROOT\AppID\{440325D6-6779-48D0-970F-02C9C7AAAAD2}

-with subkeys

HKEY_CLASSES_ROOT\WienerKVASER.OPC_SERVER

-with subkeys

HKEY_CLASSES_ROOT\WienerKVASER.OPC_SERVER.1

-with subkeys

HKEY_LOCAL_MACHINE\SOFTWARE\W-Ie-Ne-R

-with subkeys

to be continued ...

2.6 DCOM settings for OPC.

To be continued …

3 Wiener Devices Tree Configuration

The Wiener_cfg.xml file have to be placed in the same directory as WienerOPC.exe.

It contains configuration information of:

- Numbers and baud rates of CAN Nets (installed in your system) which will be used to controll Wiener devices (that CAN Nets will be used exclusively only by Wiener OPC

Server). Read Queue Length and Write Queue Length are not meningful in config file for server with KVASER interface.

- Node ID's of Wiener Devices to be controlled on each network
- ID numbers of Power Supply Channels to be controlled in each Device
- ID numbers of Temperature Sensors to be controlled in each Device
- ID numbers of Fans to be controlled in each Device

The Wiener_cfg.xml file is parsed at initialization phase of starting WienerOPC, and the "top tree structure" of server namespace is build according to the information found in it. You can not reload dynamically config file while server is running.

It is XML type file, so you can easily view it's strucure in any application supporting XML format (eg. Internet Explorer).

You can use WienerConfig.exe application to build config files. Use right mouse button on the left window tree structure elements to add/remove objects allowed on that tree level.

Fill all right window parameter fields (use scroll list to take valid values), than click OK button.

The completeness and validation of set parameters is done automatically before saving file into disk.

4 Documentation of the Server Name Space

We distinguish items with "fast" and "slow" refresh/access time.

All "fast" items (that are all changing items like measured voltage, current, ...) could be read as fast as the can bus load allowes it. (With 1 MBaud CAN speed, a repeat time of < 1 second is possible)

All "slow" items (that are values that do not change automatic, e.g. serial number, voltage channel name, nominal voltage, trip points, ...) could be read with a maximum repeat time of 5 seconds.

Our recommendation of a well designed client is: The "slow" items should be read once, and then the "fast" items should be polled as necessary.

Several Items are kept in application memory. They don't require sending CAN Messages to devices. The access to them is direct.

Address Name Space / OPC Item Name			Description	Read or Write	Type	Access Speed	Information origin
WIENER.							
	DriveError		Driver error "latch" Item. If driver error occur it is set to TRUE and timestamp is set. It never goes back to FALSE after a driver error epizode. Application should be restarted in such case.	Read	VT_ BOOL	Memory access	canERR_ DRIVERFAILED, canERR_INTERNAL, canERR_DRIVER from KVASER Canlib
	I2_Cache	UI1_Cache	"Cache" Items. Can be used to keep any user defined data during runtime.	Read Write	as visible in item names	Memory Access	OPC Client User
	UI2_Cache	BSTR_Cache					
	I4_Cache	BOOL_Cache					
	R4_Cache						
CANX.							
	BaudRate		CANbus baud rate setting	Read	VT_I4	Memory access	ncGetAttribute() from KVASER Canlib
(specific to CAN interface)	GetPortErrorCode		CAN Network Status Flags – see Appendix C	Read	VT_I4	Memory access	canReadStatus() result from KVASER Canlib
	GetPortState		CAN Network State Flags (6 low order Status Bits) – see Appendix C	Read	VT_ UI1	Memory access	canReadStatus() flags from KVASER Canlib
	GetReadQueueLenght		Actual lenght of Read Messages Queue in driver	Read	VT_I4	Memory access	canIoCtl () from KVASER Canlib
	GetWriteQueueLength		Actual lenght of Write Messages Queue in driver	Read	VT_I4	Memory access	canIoCtl () from KVASER Canlib
	FlushReadQueue		Clear Read Messages Queue in driver (TRUE) – abandon waiting Messages	Write	VT_ BOOL	Memory access	canIoCtl () from KVASER Canlib
	FlushWriteQueue		Clear Write Messages Queue in driver (TRUE) – abandon waiting Messages	Write	VT_ BOOL	Memory access	canIoCtl () from KVASER Canlib
	GetRxErrorCount		Read Receive Error Counter from driver	Read	VT_I4	Memory access	canReadErrorCounters() from KVASER Canlib

	GetTxErrorCount	Read Transmit Error Counter from driver	Read	VT_I4	Memory access	canReadErrorCounters() from KVASER Canlib
	GetOVErrCount	Read Overrun Error Counter from driver	Read	VT_I4	Memory access	canReadErrorCounters() from KVASER Canlib
	GetBusOffCount	Read Internal Counter of CANbus offs due to errors	Read	VT_I4	Memory access	Internal function
	ResetBusOffCount	Reset Internal Counter of CANbus offs due to errors (TRUE)	Write	VT_ BOOL	Memory access	Internal function
	ResetNet	Reset of CAN Network Error Counters(TRUE)	Write	VT_ BOOL	Memory access	canIoCtl() from KVASER Canlib
	SwitchNetOn	Switch Network On(TRUE) or Off(FALSE)	Write	VT_ BOOL	Memory access	canBusOn(), canBusOff() from KVASER Canlib
CrateX.						
	GetCurrentFlags	Power supply channels "switch off " flags (because of too high current)	Read	VT_ UI1	Device fast	Byte6 of IDStat from Device
(specific to Wiener Device)	GetUnderVoltFlags	Power supply channels "switch off " flags (because of sense voltage is too low)	Read	VT_ UI1	Device fast	Byte3 of IDStat from Device
	GetOverVoltFlags	Power supply channels "switch off " flags (because of sense voltage is too high)	Read	VT_ UI1	Device fast	Byte4 of IDStat from Device
	GetOverVoltageProtFlags	Power supply channels "switch off " flags (because of power supply module voltage too high (OVP))	Read	VT_ UI1	Device fast	Byte7 of IDStat from Device
	GetPowerOn	TRUE, if the crate is switched on	Read	VT_ BOOL	Device fast	Byte0/Bit0 of IDStat from Device
	OnOffCrate	Switch the crate on (TRUE), or off (FALSE). All error flags (caused by trip off) are reset.	Write	VT_ BOOL	Device fast	IDCtrl command to Device
	GetCrateStatus	Refer to IDStat message Byte1 and Byte2, to decode bit flags	Read	VT_ I2	Device fast	Byte1(low order) and Byte2(high order) of IDStat from Device
	GetNoExtInhibit	True, if the crate is switched off by the switch of the rear side of the power supply	Read	VT_ BOOL	Device fast	Byte0/Bit1 of IDStat from Device

Get Local Control	TRUE, if remote control is disabled. The state is changable with the LOCAL switch of the fan tray.	Read	VT_BOOL	Device fast	Byte1/Bit1 of IDStat from Device
GetACInLimit	Power fail(FALSE)/AC is in limit(TRUE)	Read	VT_BOOL	Device fast	Byte0/Bit2 of IDStat from Device
GetNoErrors	No errors condition (TRUE) Switched off because of any error (FALSE)	Read	VT_BOOL	Device fast	Byte0/Bit3 of IDStat from Device
GetTripIfAnyErrorEnable	Trip off if any error enabled (TRUE), or disabled (FALSE)	Read	VT_BOOL	Device fast	Byte0/Bit6 of IDStat from Device
GetNoVMESysFail	Vmebus sysfail not active (TRUE), active (FALSE)	Read	VT_BOOL	Device fast	Byte0/Bit7 of IDStat from Device
GetEEPROMChanged	Flash/eeprom data has not(FALSE)/has (TRUE) changed since last access via can bus	Read	VT_BOOL	Device fast	Byte1/Bit5 of IDStat from Device
GetEEPROMError	Flash/eeprom data checksum error (TRUE)/ no error (FALSE)	Read	VT_BOOL	Device fast	Byte1/Bit6 of IDStat from Device
GetHWWriteProtect	Write protect (TRUE), no write protect (FALSE)	Read	VT_BOOL	Device fast	Byte1/Bit7 of IDStat from Device
GetSoftstart	Device software start in progress(TRUE)	Read	VT_BOOL	Device fast	Byte1/Bit4 of IDStat from Device
GetUncompatible	Power Supply and BIN are uncompatible(FALSE)	Read	VT_BOOL	Device fast	Byte1/Bit2 of IDStat from Device
GetTempErrorFlags	External temperatures error flags	Read	VT_UI1	Device fast	Byte5 of IDStat from Device
VMESysreset	Generate a VME Subrack reset(TRUE)	Write	VT_BOOL	Device fast	IDCtrl Device command
IDStringPS	Power supply identifier	Read	VT_BSTR	Device slow	IDcfgC indexes 12,13,14,15 from Device
OperatingTimePS	Power supply operating time	Read	VT_I4	Device slow	IDcfgC index 6 from Device
SoftwareVersionCrate	CAN Crate control software version	Read	VT_BSTR	Device slow	IDcfgC index 0 from Device

SoftwareVersionPS		Power supply software version	Read	VT_ BSTR	Device slow	IDcfgC indexes 3 and 4 from Device
Fans.						
(specific to Fan Module)	ChangeFansSpeed	Change the speed of the fans	Write	VT_I2	Device fast	IDCtrl Device command
	FanSpeedX	Get the fan speed of fan X (in RPM)	Read	VT_I2	Device fast	IDFan Byte 3…8 (* 60) from Device
	GetFansOK	State of fans, Ok(TRUE), fans are broken(FALSE)	Read	VT_ BOOL	Device fast	Byte1/Bit4 of IDStat from Device
	GetTripFansBrokenEnable	Trip off if any error is enabled (TRUE) / disabled (FALSE)	Read	VT_ BOOL	Device fast	Byte1/Bit5 of IDStat from Device
	IDStringFan	Fan identifier	Read	VT_ BSTR	Device slow	IDcfgC indexes 8, 9,10,11 from Device
	MiddleSpeed	Middle fans speed (in RPM)	Read	VT_I2	Device fast	IDFan Byte 1 (* 60) from Device
	NominalSpeed	Nominal fans speed (in RPM)	Read	VT_I2	Device fast	IDFan Byte 2 (* 60) from Device
	OperatingTimeFan	Fan operating time	Read	VT_I4	Device slow	IDcfgC index 5 from Device
	SoftwareVersionFan	Software fan version	Read	VT_ BSTR	Device slow	IDcfgC indexes 1, 2 from Device
Temperature. **(specific to temperatures)**	GetExtTempErrorFlags	Temp flags errors/no error (External Temp. Probe over temperature)	Read	VT_ UI1	Device fast	Byte5 of IDStat from Device
	GetTempErrorFlags	Temp flags errors/no error (Power supply over temperature)	Read	VT_ UI1	Device fast	Byte8 of IDStat from Device
	TempLimitX	Temperature limit setting of probe X	Read /Write	VT_ R4	Device slow	IDucfgC index 8 from Device
	TempValueX	Temperature value of probe X	Read	VT_ R4	Device fast	IDtemp Byte 1…8 from Device

ChannelX. **(specific to power supply channels)** **(see Exponent explanation)**	TempWarningX	Temperature warning setting of probe X **X – 0…7**	Read /Write	VT_ R4	Device slow	IDucfgC index 7 from Device
	CurrentLimitSetPoint	Current limits settings	Read /Write	VT_ R4	Device slow	IDucfgC index 1 from Device
	CurrentValue	Actual current value of channel X	Read	VT_ R4	Device fast	IDvcX from Device
	MinCurrentCompSetPoint	Minimum current compare settings	Read /Write	VT_ R4	Device slow	IDucfgC index 4 from Device
	OverCurrentCompSetPoint	Over current compare settings (Over current trip off value)	Read /Write	VT_ R4	Device slow	IDucfgC index 5 from Device
	OverVoltCompSetPoint	Over voltage compare settings (Sense Lines)	Read /Write	VT_ R4	Device slow	IDucfgC index 3 from Device
	OverVoltProtection	Over voltage compare settings (Power Supply Output)	Read /Write	VT_ R4	Device slow	IDucfgC index 6 from Device
	UnderVoltCompSetPoint	Minimum voltage compare settings	Read /Write	VT_ R4	Device slow	IDucfgC index 2 from Device
	FineAdjust	Adjustment of the voltages in small steps (DAC setting)	Read /Write	VT_I2	Device slow	IDucfgC index 9 from Device
	VoltageSetPoint	Voltage setpoint	Read /Write	VT_ R4	Device slow	IDucfgC index 0 from Device
	VoltageValue	Actual voltage value of channel X	Read	VT_ R4	Device fast	IDvcX from Device

Exponent explanation: Each power supply channel have two exponent values associated with. One of them is for current values, second one to voltage.

That exponents are read from all devices connected to Server's CAN Networks during application initialization. If any of that values are not known due to communication problems with some channels during initialization, OPC Server will try to read proper exponent, when it is needed to send new setpoint value or interpret the actual value in corresponding channel. That mechanism can result in substantial CAN traffic overhead if all IDucfgC which can bring desired exponent value are not implemented (some old versions of Wiener firmware can cause such problem).

APPENDIX A: Sample Server Configuration File

The following file shows the configuration for one crate (with crate ID 1). To access all crates, the "<CRATE_BRANCH CrateID="2"> ... <CRATE_BRANCH CrateID="126">

```
<?xml version="1.0"?>
<ROOT name="WIENER" version="1.1">
<PORT_BRANCH PortID="CAN0" BaudRate="100000" ReadQueueLenght="0"
WriteQueueLenght="0">

<CRATE_BRANCH CrateID="1">
<CHANNEL_BRANCH ChannelID="0"/>
<CHANNEL_BRANCH ChannelID="1"/>
<CHANNEL_BRANCH ChannelID="2"/>
<CHANNEL_BRANCH ChannelID="3"/>
<CHANNEL_BRANCH ChannelID="4"/>
<CHANNEL_BRANCH ChannelID="5"/>
<CHANNEL_BRANCH ChannelID="6"/>
<CHANNEL_BRANCH ChannelID="7"/>
<FANS_BRANCH>
<Fan FanID="1"/>
<Fan FanID="2"/>
<Fan FanID="3"/>
<Fan FanID="4"/>
<Fan FanID="5"/>
<Fan FanID="6"/>
</FANS_BRANCH>
<TEMPERATURES_BRANCH>
<Temp TempID="1"/>
<Temp TempID="2"/>
<Temp TempID="3"/>
<Temp TempID="4"/>
<Temp TempID="4"/>
<Temp TempID="5"/>
<Temp TempID="6"/>
<Temp TempID="7"/>
<Temp TempID="8"/>
</TEMPERATURES_BRANCH>
</CRATE_BRANCH>

... here it is possible to add additional crate branches

</PORT_BRANCH>
</ROOT>
```

APPENDIX B: Server Specific Error Codes

	OPC Server Error Code	Description
KVASER CAN Interface related		
Error Code		
canERR_DRIVERFAILED	0xE0048620	DeviceIOControl failed
canERR_INTERNAL	0xE0048621	Internal error in the driver
canERR_NOCARD	0xE0048622	The card was removed or not inserted
canERR_NOTFOUND	0xE0048623	Specified hardware not found
canERR_NOHANDLES	0xE0048624	Can't get handle
canERR_NOMEM	0xE0048625	Out of memory
canERR_NOCHANNELS	0xE0048626	No channels avaliable
canERR_INIFILE	0xE0048627	Error in the ini-file
canERR_DRIVER	0xE0048628	CAN driver type not supported
canERR_PARAM	0xE0048629	Error in parameter
canERR_TIMEOUT	0xE004862A	Timeout ocurred
canERR_HARDWARE	0xE004862B	Some hardware error has occurred
canERR_TXBUFOFL	0xE004862C	Transmit buffer overflow
canERR_NOMSG	0xE004862D	No messages available
canERR_INVHANDLE	0xE004862E	Handle is invalid
canERR_NOCONFIGMGR	0xE004862F	Can't find required config s/w (e.g. CS/SS)
canERR_DRIVERLOAD	0xE0048630	Can't find/load driver
canMSGERR_BIT0	0xE0048631	Send dominant, read recessive
canMSGERR_BIT1	0xE0048632	Send recessive, read dominant
canMSGERR_CRC	0xE0048633	CRC error
canMSGERR_FORM	0xE0048634	FORM error
canMSGERR_STUFF	0xE0048635	STUFF error
canMSGERR_SW_OVERRUN	0xE0048636	Software buffer overrun
canMSGERR_HW_OVERRUN	0xE0048637	Hardware buffer overrun
canERR_DYNALOAD	0xE0048638	Can't find requested DLL
canERR_DRIVERLOAD	0xE0048639	DLL seems to be wrong version
canERR_DYNAINIT	0xE004863A	Error when initializing DLL
canERR_REGISTRY	0xE004863B	Error in the Registry
canERR_LICENSE	0xE004863C	The license is not valid
canERR_NO_ACCESS	0xE004863D	Access denied
canERR_NOTINITIALIZED	0xE004863E	Lib not initialized
canSTAT_BUS_OFF	0xE004863F	Net is stopped, write frame can not proceed
canERR_RESERVED	0xE0048640	Unknown errors - reserved codes
out of errors ranges	0xE0048641	Unknown error code
	0xE0048642	Net not initialized – handle not found
WIENER Device and Configuration File related		
Error Code		
WIENER_E_CMDLINE	0xE0048501	Unrecognized command line switch
WIENER_E_XMLPARSE	0xE0048502	Config file - XML format

		parsification failed
WIENER_E_NOXMLTAG	0xE0048503	Config file - expected XML Tag not found
WIENER_E_XMLATTRVAL	0xE0048504	Config file - invalid XML Attribute value
WIENER_E_XMLATTRNAME	0xE0048505	Config file - invalid XML Attribute name
WIENER_E_BADXMLTAG	0xE0048506	Config file - Bad XML Tag found
WIENER_E_PROTOCOL_NOT_RESPONSE	0xE0048700	Message not recognized as a device response
FAILMESS_WRITEPROTECT	0xE0048701	Fail message - trying to program write protected data
FAILMESS_BADVALUE	0xE0048702	Fail message - value not allowed(out of boundaries)
FAILMESS_UNDEFINED_COMMAND	0xE0048703	Fail message - undefined command
FAILMESS_COMMAND_NOT_SUPPORTED	0xE0048704	Fail message - command not supported by an existing hardware
FAILMESS_BAD_BYTE_COUNT	0xE0048705	Fail message - byte count not allowed
FAILMESS_DATA_OVERRUN	0xE0048706	Fail message - data overrun
FAILMESS_EEPROM_CHECKSUM	0xE0048707	Fail message - hardware error(EEPROM checksum not OK)
FAILMESS_EEPROM_ACCESS	0xE0048708	Fail message - hardware error(unable to access EEPROM data)
FAILMESS_NOT_DEFINED	0xE0048709	Fail message - Status code of message not known
DEVICE_RESPONSE_TIMEOUT	0xE004870A	Device didn't send back a response message during timeout period
PARAMETER_EXPONENT_NOT_KNOWN	0xE004870B	Exponent value for requested write parameter not known, read any setpoint
FAILMESS_ILLEGAL_CHANNEL	0xE004870C	Illegal channel number
FAILMESS_COMMAND_NOT_EXECUTED	0xE004870D	Command not executed - power supply is in Local Control Mode

APPENDIX C: KVASER CANLIB Status Flags

Status Flag	Flag Mask	Meaning
canSTAT_ERROR_PASSIVE	0x00000001	The circuit is error passive
canSTAT_BUS_OFF	0x00000002	The circuit is Off Bus
canSTAT_ERROR_WARNING	0x00000004	At least one error counter > 96
canSTAT_ERROR_ACTIVE	0x00000008	The circuit is error active.
canSTAT_TX_PENDING	0x00000010	There are messages pending transmission
canSTAT_RX_PENDING	0x00000020	There are messages in the receive buffer
canSTAT_TXERR	0x00000080	There has been at least one TX error
canSTAT_RXERR	0x00000100	There has been at least one RX error of some s
canSTAT_HW_OVERRUN	0x00000200	The has been at least one HW buffer overflow
canSTAT_SW_OVERRUN	0x00000400	The has been at least one SW buffer overflow
canSTAT_OVERRUN		For convenience. This flag is the logical or between canSTAT_SW_OVERRUN and canSTAT_HW_OVERRUN.

APPENDIX D: Version History

Version 1.0.0.0 alpha (internal version only)

- ✓ Windows NT, National Instruments CAN-hardware.

Version 1.0.0.1 beta (1.12.2002)

- ✓ Windows 2000, National Instruments CAN-hardware.

Version 1.0.0.2 beta (7. Feb. 2003)

- ✓ Windows 2000, KVASER CAN-hardware.
- ✓ Local Control of the Fan-Tray implemented.
- ✓ Adjustment of some Timeouts.

Version 1.0.0.3.beta (17. Feb. 2003)

- ✓ New fail message error codes implemented.

Version 1.0.0.4 (internal version)

- ✓ CANbus automatic restart after bus off due to network errors. GetBusOffCounter and ResetBusOffCounter namespace Items added to follow a number of such restarts.
- ✓ Project migration to Softing Toolbox v.3.10 OPC Data Access Toolkit
- ✓ Removing all MFC dependent objects – Softing Toolbox corresponding objects used instead. MFC library not loaded into application any more
- ✓ Registry Key Entries redefinition
- ✓ Getting LoLimit, HiLimit and Exponent values for each Channel Voltage and Current Items from Wiener devices during OPC Server startup (General Call Arb.ID 127 used)
- ✓ "Trace to logfile" setting command line options added
- ✓ Registry Key Entries enhancement for trace options setting

Version 1.0.0.5 (10 July 2003)

- ✓ Getting an Exponent value before sending a request CAN message for Item with Exp.is not initialised
- ✓ Referencing for OPC Requests improved (not allowing to destroy object before taking response completion or timeout action)
- ✓ Error code setting for Fail Message response (write to device request) corrected
- ✓ Softing Toolbox upgraded to version 3.12
- ✓ Read Item Property pointed to Cache values instead of getting them from the device
- ✓ Checking access rights during Group Refresh
- ✓ "Cache Items" added for better testing during OPC Compliance Tests. User defined data can be kept in runtime.

Version 1.0.0.6 (10 September 2003)

- ✓ Boolean status values representing TRUE for ***GetNoErrors***, ***GetNoExtInhibit, GetNoVMESysfail*** items represented by 1 (not other non-zero values)
- ✓ ***GetVoltageFlags*** VT_I2 item splitted into two VT_UI1 items: ***GetUnderVoltFlags*** and ***GetOverVoltFlags***
- ✓ Problem of crashes during Stress Tests (when device was responding with fail messages) resolved
- ✓ New GUID Registry Key value defined